01584 Grundpraktikum Programmierung

Einführung in das Programm

Hans Peter Müller
Matrikel-Nr. 3274969
WS 2021/22

Inhalt

Inhalt **2**

1. Einleitung **3**
    1.1. Petrinetze 4
    1.2. Erreichbarkeitsgraphen 4
    1.3. Icons 4

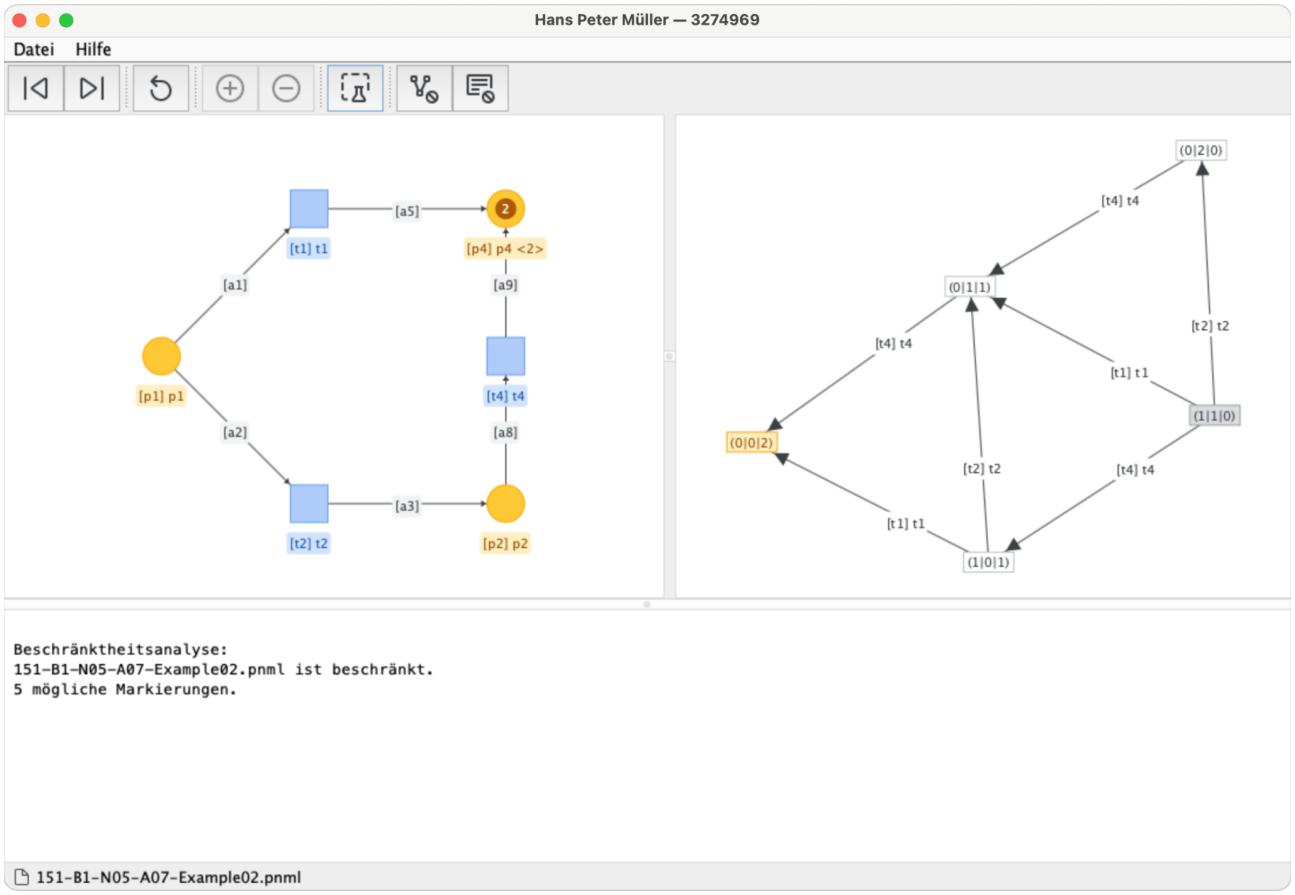
Beschreibung der Programmstruktur **4**

Beschreibung des Beschränktheits-Algorithmus **5**

1. Einleitung

Diese Einführung in das Programm gibt einen Überblick über den Funktionsumfang sowie die Architektur meines Petrinetz Programms, das im Rahmen des Grundpraktikums Programmierung entstanden ist. Abschließend wird der von mir entwickelte Beschränktheitsanalyse-Algorithmus vorgestellt und anhand von Pseudocode die Vorgehensweise des Algorithmus erklärt.

Der Funktionsumfang meines Programms konzentriert sich auf die Kernanforderungen der Aufgabenstellung: Das vorgestellte Programm kann also Petrinetz-Dateien laden und visualisieren; Benutzer können mit dem Petrinetz interagieren und zum Beispiel Transitionen schalten sowie neue Marker setzen; ein Erreichbarkeitsgraph baut sich sukzessive auf und die Beschränktheitsanalyse ist sowohl für einzelne Petrinetz-Dateien wie auch für mehrere via Stapelverarbeitung möglich. Zusätzlich habe ich alle im Programm verwendeten Icons selbst entworfen und umgesetzt.



Hauptansicht mit Petrinetz, Erreichbarkeitsgraph und Ergebnis einer Beschränktheitsanalyse

1.1. Petrinetze

- legende

1.2. Erreichbarkeitsgraphen

- legende
- Markierung
- Wurzel
- m
- m’
- pfad

1.3. Icons

- Umgesetzt wurden nur die Grundanforderungen
- [Screenshot graphendarstellung]
- [Screenshot batchverarbeitung]
- Icons wurden selbst gestaltet

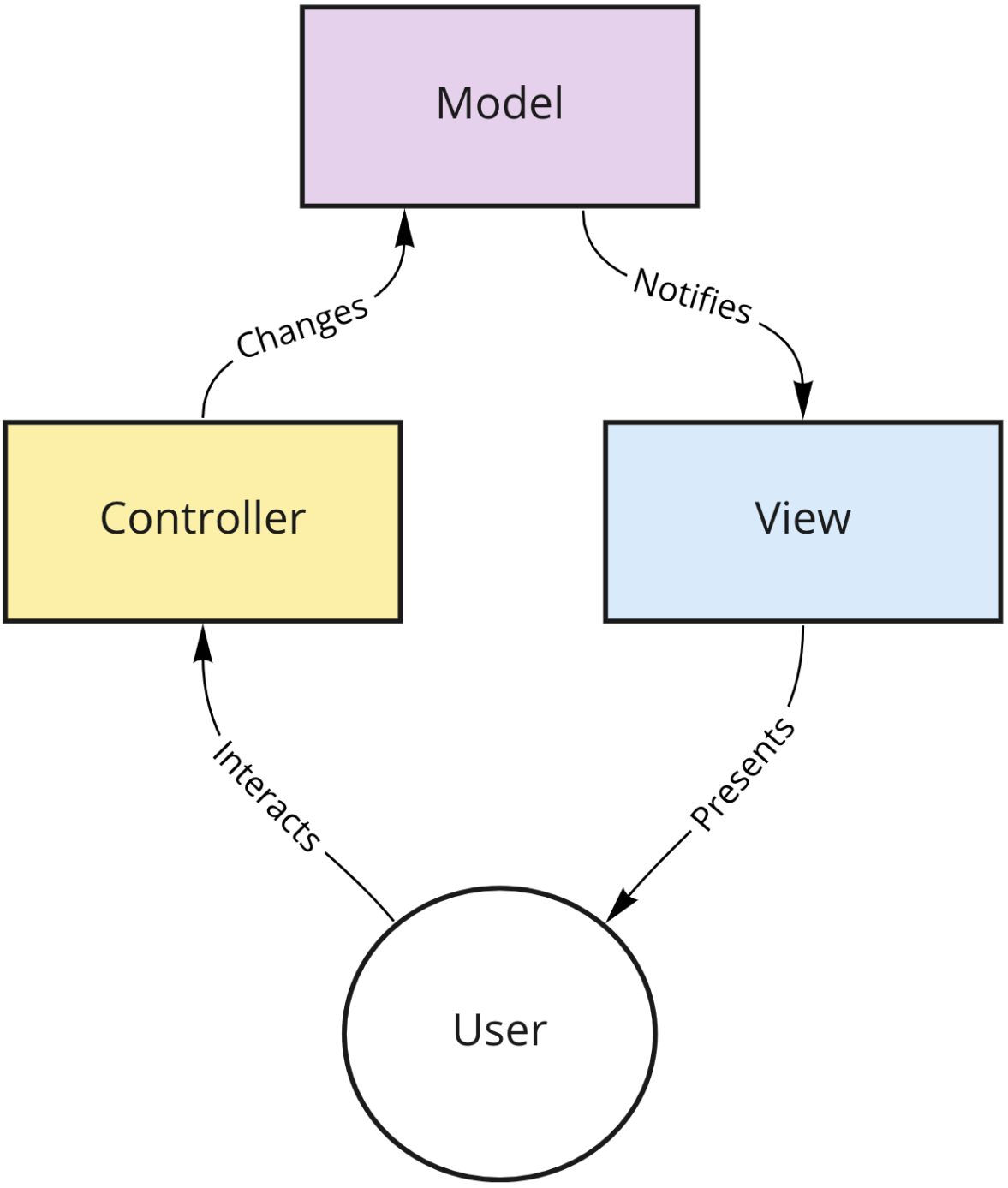
Beschreibung der Programmstruktur

- MVC Architektur
    - Aufgabenbeschreibung, separation of interests
    - Controller kümmern sich nur um **user**aktionen, views um die Präsentation und Models um die Daten.
    - Modelle daten Views via Observerpattern up
    - Controller installieren listener und reagieren auf userinput. Sie aktualisieren Models, greifen aber so gut wie nie direkt in Views ein.
    - Alle Controller/Models/Views in ein package gesteckt, um die struktur einfach zu halten
    - utils Package nimmt alle Klassen auf, die nicht in das Schema passen
    - [Bild Klassenstruktur]
- Die Main Klasse initiiert alle Objekte direkt beim Start
    - Model, View(model), Controller(model, view)
    - Die Objekthierarchie ist absichtlich flach gehalten, wenig geschachtelte Klassen (Ausnahmen sind kleine Komponenten wie Listener, Contextmenüs, und Buttons
    - [Bild Objekthierarchie]

- Die Drei Hauptmodels
    - Kurzbeschreibung aller drei
    - Petrinetz
        - Idee: hält sich nahe an dem, was der Parser liefert, da auch Graphstream genau diese Klassen erwartet
        - Places
        - Transitions
        - Arcs
        - Attribute
        - Wichtigste Methoden
    - rGraph
        - Idee: hält Markings und arcs zwischen den markings
        - Markings
        - TransitionArc
        - Wichtigste Methoden
    - Filesystem
        - Hilfsmodell um die Arbeit mit dem Betriebssystem wegzuabstrahieren
        - Wichtigste Methoden
        - Fehlerbehandlung
- Die wichtigsten Views und ihre Controller
    - mainFrame
    - toolbar
    - Petrinetz
    - RGraph
    - TextArea

Beschreibung des Beschränktheits-Algorithmus

- Ist alt utility-klasse realisiert
- der algorithmus erwartet ein petrinetzmodell und stellt die ergebnisse der analyse im objekt als öffentliche attribute bereit (vorsicht keine implementierung)
- vorgehen beschreiben
    - breitensuche durch das petrinetz
    - In der aktuellen Markierung wird überprüft, welche Transitionen schaltbar sind
    - diese werden rekursiv in einer breitensuche weiter abgeklappert
    - Abbruchkriterien sind die Beschränktheitskriterien (m, m') sowie Kreise im Graphen
- pseudocode


Model
Changes
Notifies
Controller
View
Interacts
Presents
User